New Uses of Multicast

All nodes working on one problem

Mon, Sep 15, 2003

The search for a match on an analog name is one use of multicast-based searching. This note is meant to explore the possibility of others.

The built-in support for analog name searching accepts only the first reply, in case more than one node reached via multicast addressing finds a match. It should be possible to write a page application to accept an analog name, send it as a Classic request message from a source port other than 6800, and receive replies from multiple nodes. This would provide the new feature of listing all nodes reporting a given 6-character name. In practice, this is not so unusual, especially for HLRF nodes, as many names are common between Booster, MI, and Tevatron. At the Acnet level, of course, the names do not conflict, as Acnet actually uses 8-character names such as "`B:MD05V `" or "`M:MD05V `".

In the case of name searches, it is easy to identify the node number of the system responding because it is included in the 4-byte data that is returned. Consider using a similar approach for other searches. It is still easy to notice multiple replies, but the data in the reply is unlikely to include a node number. A pseudo node# instead for each reply message is delivered to the client by the system. These pseudo node numbers could be translated using the relatively recent listype 95, which was designed for the purpose.

Other idents include the target node number, unless they can also permit a multicast node number (`MNN`) in the node number field. There is code that scans through the list of idents, changing each Acnet node number that matches the local Acnet node number into the local native node number. This is done so that the ptr-type routine loops that scan through the ident array need only check for a match on the native node number. This can be changed so that it also looks for a valid (acceptable by the local node) `MNN`. With this change, it should be possible to perform other kinds of searches, not only that of an analog channel name.

Suppose we only allow a single ident (and a single listype) in a request of this sort. When a reply message is received from a node, it is therefore clear to what the reply data refers. This same restriction is enforced for analog channel name lookup listype 19 for the same reason. The destination node will likely be some `MNN`. The single ident would also have the same `MNN`. The pre-scan logic would replace the `MNN` with the local node number to satisfy the ptr-type routine. So, even though the multicast request was sent to many nodes, each node that receives the multicast message will decide that it is being addressed.

Explore some possible uses for this kind of request. One could ask for the reading of a given analog channel, using listype 0, which would likely result in a reply from each node that received the request message. This does not seem too interesting. The same is true for many of the listypes that use an analog channel ident.

One could ask for a specific global variable. A reply would ensue from all nodes that receive the request. (Of course, if one knew ahead of time that a certain set of nodes were to be targeted, one could merely send a request with that many idents.) But consider a request for the native node number global. The replies would include the node number of each node receiving the request. With the above refinement, this is a nice way to produce a list of node numbers answering a particular `MNN`. Indicate `09FB` in the single ident. Replies from all nodes listening to `09FB` would ensue. One would have a count and a complete list of nodes in the `09FB` "family."

Listype 76 allows asking for a particular named file found in CODES. One could find out in what nodes of a project a given program file is housed. If the offset field in the ident is –1, the matching CODES table entry is returned.

Listype 86 searches for a matching LA name, returning one or more LATBL entries. This allows finding where a given LA is used. In this case, the LA name is the internal ptr, so that the read-type routine performs the search through the LATBL for one or more matches on the name to return in the reply buffer. If no matches are found, the reply buffer is cleared. This is harder to implement, since only a ptr-type routine can refuse to return a reply; a read-type routine must generate the reply data for the caller to return.

Until now, the only listype for which a reply might not be returned at all, depending on the ident, is the analog name search. In all other cases in which the ptr-type routine finds a match on the node number part of the ident, a reply ensues. The data may be zero, but there is a reply. This means that looking for a given program file name, for example, may result in a huge number of replies, if the request is sent to 09F9. If there were a way to keep this from happening, it would be better. It would mean that a listype whose ptr-type routine performs a search would need to determine whether the original node number in the ident was an MNN. If it was, and if the search failed, it should ignore the request. A multicast request, when the node number does not match the local node number for any of the idents, is ignored.

Implementing such a scheme will also require a suitable solution for receiving many replies, which for an IRM means a larger area used for an ethernet receive buffer, which itself implies having solved the problem of the NETFRAME records tagged as denoting a received or transmitted datagram. Another note, *Network Frames Addition*, covers this.

The current logic in the Classic task for processing a request message is to check the server bit first. If it is *not* set, then call PReqData to process the request; otherwise, if the destination for the request message was *not* multicast, then process it as a server request by invoking ReqDataS. Any server request that was received via multicast is ignored.

The idea here is that a request suitably formed from a single ident and listype could prompt a reply from a node that is included within a multicast node number domain. For three listypes, the reply under these conditions might be missing if the search fails. The necessary ingredient in the protocol should include a MNN in the ident.

A different tack

Suppose a new listype is designed especially for name look-ups. The ptr-type routine for this case would only return a reply if a match is found for whatever search is implied. Only a single ident and listype would be used in such a request. It could be either multicast or unicast. The node# field of the ident could be zero, meaning that the receiving node is being addressed. Suppose the ident is of this format:

node
type
name[8]

The length of the ident should adhere to some fixed maximum. Perhaps it could be 8 bytes, since no names are longer than that; in that case, the ident length would be 12 bytes. The type word would specify the kind of search desired, meaning that the requester knows that code. The name types under active consideration here are:

Type	*Size*
0. Analog name	6
1. LATBL name	4
2. CODES file name	8
3. DSTRM name	8

The return data would be a node# and an entry#, adding up to 4 bytes. In the case of a LATBL name, there can be more than one entry. In that case, asking for a multiple of 4 bytes would allow space to receive additional matches. Every other word would be the same (nonzero) node number. It is easy to count the returns in the buffer.

The advantage of this approach is that it more naturally fits into the present scheme of support. All searching must be done by the ptr-type routine, since that is a means of refusing any reply at all.

If the node# in the ident matches the local node#, then it may be better to return a reply even if the search fails, since the request was presumably sent only to that node. But if the ident# is zero, then the reply should be returned only if a match is found. Thus, one can interrogate a single node and receive an immediate reply whether or not a match is found. But when sending the request via multicast, one would not want a reply except from a node (or nodes) that find a match.

For a client, it may be best to use the new listype under the following rules:

Only one ident per request
Request sent with multicast (ident node#=0) or unicast (ident node#=target).

Again, if the request is sent unicast, a response will come right away, the reply data being the node# and entry# for the case of finding a match on the name, and 4 bytes of zero otherwise. If the request is sent multicast, a reply will ensue only for a match.

There will still need to be special code in PSLN for this new listype that avoids scanning the node# fields to count how many are local idents. The ident type# is used for this.

Since the D0 protocol is no longer supported, perhaps the 16-character name lookup listype (#55) could be used for this new case. Probably no clients use it the old way, anyway. Even if they do, no reply will ensue because it will be an invalid type code. For new clients, no reply will ensue from old nodes because the type code will make it appear to be an invalid name.

If new type codes need to be supported under the new listype, the underlying code will have to be modified, of course. But that's ok.

These 4 cases for searches involve different logic. The analog name search involves a call to NTLookup. The LA name lookup should search the online copy of LATBL, if there is one. The file name case must require a search of the CODES table. (The only optimization for this in the PowerPC case was to remember where a given name was found. For this new searching listype, there is no such context memory.) For the DSTRM name case, one must search the nonvolatile DSTRM table, as there is no online copy.

The new case can be handled by all the usual data request support, if the request is unicast. But if it is multicast, it must be supported in a special way, since more than a single reply may ensue, which the underlying system code does not handle.

In this new way, there is no need to support multicast node numbers in the ident field. A zero value will be adequate, and it should be used only for the case of multicasting the request. A unicast request with a zero node# will not cause any reply.

Client page application

Suppose all this were installed. What kind of user interface would make it usable?

```
      0123456789012345678901234567890 1
0     X NAME SEARCHES   xx/xx/xx xxxx
1     NODE<    >
2     ANLG<      >    LAPP<    >
3     FILE<        >  DSTR<        >
4     NODE:INDX  NODE:INDX  NODE:INDX
5—14
15
```

This layout allows for three columns of 10 replies each. A listing output could permit more. The entered NODE number may be an individual node# or a multicast node#. If it is a multicast node#, the ident that is sent will have its node field set to zero. All four types of names have separate fields, so the type code is determined indirectly by the cursor location when the keyboard interrupt (or mouse-click) occurs.

Post-implementation

The page layout was only slightly modified from the above plan in order to allow for some additional status and diagnostics.

```
      0123456789012345678901234567890 1
0     X NAME SEARCHES   xx/xx/xx xxxx
1     NODE<    >      DT=       N=
2     ANLG<      >    LAPP<    >
3     FILE<        >  DSTR<        >
4     NODE:INDX  NODE:INDX  NODE:INDX
5—14
15
```

The DT field shows the elapsed time in milliseconds from sending the request message to the last reply received. The N field merely shows the number of replies received.

At present there is no listing output option, but up to 30 results can be displayed in response to a single one-shot request for name lookup. For the LApp case, up to 30 matches can be captured from a single node. For the other three search types, only a single result from each node can be entertained. (This is also true of the system level support for this new listype.)

One wrinkle with this page application is that it cannot find matches in the local node, because the Classic protocol support at the system level ignores all messages received by the local node that were sent from the local node. One modification to help alleviate this limitation would be, for the case of the target node matching the local node#, to call ReqData to query the local node, then call Collect to retrieve the results. Even better would be to recognize when a target MNN is one listened to by the local node and fold in the local request for that case as well. (This modification was added Sep 12, 2003.)

To use the program, enter the target node, either specific or multicast, in the NODE field. (One need not interrupt here; the node number field will be read for any action selected.) Interrupt with the cursor in any of the four fields of interest, after having entered the name to be searched. The extent of each field is all the text comprising the field. The program knows that

you want to search for File names, for example, when you interrupt with the cursor in the File name field.

The searches only look for a match on the name field of the relevant table entries. The `LApp` search, for example, will find a match even if the `LATBL` entry is disabled and even if there is no such local application in the memory file system. It only cares if a `LATBL` entry contains the name that is sought.

We notice that sometimes a multicast datagram does noes not seem to reach all nodes in the multicast group, necessitating a retry to get a valid result. If there is no resolution to this problem, we may have to build the retry into the page application logic. In the meantime, not having it built-in may allow us to get a better handle on when a retry is required.

The page application `NAME` consists of 680 lines of Pascal source code and compiles into an executable file of about 4K bytes.

Internal system details

New ptr-type #39 routines were written named `NSearch`, called by `PReqDGen`, and `RSearch`, called by `ReqDGenP`. These routines replaced the old `LongName` routines. The ident type #15 has a new 12-byte length, replacing the old 18-byte length. These changes were done when redefining the meaning of listype #55, which once supported 16-character D0 device name lookups. (The D0 protocol is no longer used nor supported.)

A new read-type #30 routine was written named `RdNSrch`. It simply examines the (single) internal ptr to decide whether to invoke the new routine `RdLATBLI`, used for the `LApp` case, or to invoke `ReadCh`, which is also used by read-type #5, in which the internal ptr actually contains the data to be returned, the result of the search performed by the ptr-type routine during request initialization. (Read-type #5 is used by listype #19, which searches the `ADESC` table entries for analog channel names in its ptr-type #33 routine.)

Summary

Considering what additional uses could be made of multicast addressing resulted in broadening the possibilites for supporting name lookups/searches. A single listype support was designed to support searches for the 4 types of names used in the system: analog channel names, local application instances, nonvolatile memory file names, and data stream names. A new page application called `NAME` was implemented as a client for this support. When using multicast addressing to target many nodes, the replies appear instantly.